Apple II
Technical Notes

Developer Technical Support

Apple IIGS
#38: List Controls in Dialog Boxes

Revised by: C.K. Haun September 1990
Written by: Keith Rollin, Dave Lyons & Eric Soldan May 1988

This Technical Note describes how to include a list control into a dialog box. Sample APW C source code is included.

Changes since March 1990: Changed input parameter definition for `myFilterProc` from long pointer to word pointer.

The need to put a list control into a dialog box is obvious. The Print Manager does it. The Font Manager does it. You may want to use one in your own application to manage a list of data base fields or spreadsheet functions. However, performing the task is not as obvious as the need.

Given the features of TaskMaster in System Software 5.0, it is now much easier to emulate a modal dialog in a normal window. If you need to add a list control to a modal dialog, you should seriously consider emulating a modal dialog with a normal window instead of using the Dialog Manager. If you use the Dialog Manager, the following procedure and sample C fragment illustrate the technique necessary for adding a list control.

Note that only **one** list control is allowed in a modal dialog. If you need more than one, the Dialog Manager cannot help you—create a normal window instead.

Individual Steps

Basically, there are three check-off items for putting a list control into a dialog box:

1. You must install the list explicitly into the dialog box yourself. This should be done **after** you have created the dialog box with a call to `NewModalDialog` or `GetNewModalDialog`. Do **not** install it as a `UserItem` or `UserCtlItem`. Installing it as a `UserItem` would cause the Dialog Manager to place an invisible custom control over the list, preventing later use of `FindControl` to manage it. Installing the list as a `UserCtlItem` does not allow the list control to be properly initialized.

   Note: After you add the list control, you must **not** add any more dialog items.

   ```
   InitValues()
   ```

```
{
    /* Get a Full Screen, invisible dialog window with only
       a Quit button in it*/
    myDialog = GetNewModalDialog(&PrintDialog);
```

```
    /* Add this List Control ourselves */
    myListHndl = CreateList(myDialog,&myList);

    /* Get the handle for the Scrollbar Control */
    listScrollHandle = (**myListHndl).ctlListBar;

    /* Save and Zero out the RefCons */
    listRefCons = GetCtlRefCon(myListHndl);
    scrollRefCons = GetCtlRefCon(listScrollHandle);
    ZeroRefCons(); /* This is explained below in item #3 */

    /* Now show the dialog box */
    ShowWindow(myDialog);
}
```

2. Because the list control is not a dialog item, a custom `FilterProc` must be installed for `ModalDialog` to test for mouse-down events. Pass the address of this routine (with the high bit set so that default handling of items is in effect) when you call `ModalDialog`.

```
pascal Word myFilterProc(theDialog, theEvent, theItem)
    GrafPortPtr    theDialog;
    EventRecord    *theEvent;
    word           *theItem;

{
    CtlRecHndl  tHandle;

    if ((*theEvent).what == mouseDownEvt) {
        FindControl(&tHandle,(*theEvent).where,theDialog);
        if ((tHandle == myListHndl) || (tHandle == listScrollHandle)) {

            /* Set the RefCons back to the way the list manager likes them */
            RestoreRefCons();
            TrackControl((*theEvent).where,(LongProcPtr) -1, tHandle);
            ZeroRefCons();

            /* Tell the Dialog Manager that we handled this event */
            return(true);
        }
    }
    /* We didn't do anything, so return false to get Dialog Manager
        to handle this event */
  return(false);
}
```

3. The Dialog Manager uses the `RefCon` field of its items (all of which are installed as controls). Unfortunately, the List Manager also uses the `RefCon` field for its own purposes. This shared use means that a judicious juggling of those values is required. This juggling is the reason for the two routines `RestoreRefCons` and `ZeroRefCons` used above.

```
/* Zero out the RefCons for the Dialog Manager */
ZeroRefCons()
{
    SetCtlRefCon(0,myListHndl);
    SetCtlRefCon(0,listScrollHandle);
}
```

```
/* Restore the RefCons for the List Manager */
RestoreRefCons()
{
    SetCtlRefCon(listRefCons,myListHndl);
    SetCtlRefCon(scrollRefCons,listScrollHandle);
}
```

Note: Because the Dialog Manager currently uses the `RefCon` to keep track of which dialog item is identified with which particular control, zeroing the `RefCon` fields can cause a little confusion. Specifically, those who would like to do `GetFirstDItem` from within a Standard File call may get a zeroed `RefCon` as a result. This is true for Standard File 3.0 and later (System Software 5.0), as this is the first implementation of Standard File to use the List Manager.

Putting It All Together

Here are most of the pieces put together. `InitTools` and `ShutDownStuff` routines have been omitted, but they are straightforward.

```
char              **y,*z;
GrafPortPtr       myDialog;
ListCtlRecHndl    myListHndl;
CtlRecHndl        listScrollHandle;
long              listRefCons, scrollRefCons;

#define Quit      ok

char  quitStr[] = "\pQuit";

ItemTemplate quitButton =  {
           Quit,
           140,450,154,590,
           buttonItem,
           quitStr,
           0,
           0,
           NULL};

DialogTemplate PrintDialog = {
           30,20,190,620,
           false,
           0,
           &quitButton,
           NULL};

char string1[] = "String1";
char string2[] = "String2";
char string3[] = "String3";
char string4[] = "String4";
char string5[] = "String5";
char string6[] = "String6";
char string7[] = "String7";
char string8[] = "String8";
```

```
MemRec  myMembers[8] = {
           string1, 00,
           string2, 00,
           string3, 00,
           string4, 00,
           string5, 00,
           string6, 00,
           string7, 00,
           string8, 00};

ListRec  myList = {
           40,175,102,400, /* Enclosing Rectangle */
           8,              /* Number of List Members */
           6,              /* Max Viewable members */
           3,              /* Bit Flag */
           1,              /* First member in view */
           NULL,           /* List control's handle */
           NULL,           /* Address of Custom drawing routine */
           10,             /* Height of list members */
           5,              /* Size of Member Records */
           (MemRecPtr)myMembers,/* Pointer to first element in MemRec[] */
           NULL,           /* Becomes Control's refCon */
           NULL            /* Color table for list's scroll bar */
           };

/* ************************* */

main()
{
       word what;

       InitTools();         /* initialize tools */
       InitValues();        /* Get dialog box. Install List control */
       do {
           what = ModalDialog((WordProcPtr)((long)myFilterProc | 0x80000000));
       } while (what != Quit);
       ShutDownStuff();
}

pascal Word myFilterProc(theDialog, theEvent, theItem)
       GrafPortPtr    theDialog;
       EventRecord    *theEvent;
       word           *theItem;

{
       CtlRecHndl     tHandle;

       if ((*theEvent).what == mouseDownEvt) {
           FindControl(&tHandle,(*theEvent).where,theDialog);
           if ((tHandle == myListHndl) || (tHandle == listScrollHandle)) {

               /* Set the RefCons back to the way the list manager likes them */
               RestoreRefCons();
               TrackControl((*theEvent).where,(LongProcPtr) -1, tHandle);
               ZeroRefCons();

               /* Tell the Dialog Manager that we handled this event */
               return(true);
           }
       }
       /* We didn't do anything, so return false to get Dialog Manager
          to handle this event */
  return(false);
}
```

```
/* Zero out the Refcons for the Dialog Manager */
ZeroRefCons()
{
       SetCtlRefCon(0,myListHndl);
       SetCtlRefCon(0,listScrollHandle);
}

/* Restore the Refcons for the List Manager */
RestoreRefCons()
{
       SetCtlRefCon(listRefCons,myListHndl);
       SetCtlRefCon(scrollRefCons,listScrollHandle);
}

InitValues()
{
       /* Get a Full Screen, invisible dialog window with only a Quit button in it*/
       myDialog = GetNewModalDialog(&PrintDialog);

       /* Add this List Control ourselves */
       myListHndl = CreateList(myDialog,&myList);

       /* Get the handle for the Scrollbar Control */
       listScrollHandle = (**myListHndl).ctlListBar;

       /* Save and Zero out the RefCons */
       listRefCons = GetCtlRefCon(myListHndl);
       scrollRefCons = GetCtlRefCon(listScrollHandle);
       ZeroRefCons();

       /* Now show the dialog box */
       ShowWindow(myDialog);
}
```